

REMARKS

Reconsideration and allowance of the above identified patent application are hereby requested. Claims 1-9, 21-29, and 37-58 are now in the application with claims 1, 21, and 45 being independent. The Office's rejections are respectfully traversed.

Rejection Under 35 U.S.C. §102

Claims 1-9, 21-29, and 35-57 stand rejected under 35 U.S.C. §102(e) as allegedly being anticipated by U.S. Patent No. 5,907,326 to Atkin et al. The Office's contentions are respectfully traversed.

Claim 1 recites (underlining added for emphasis) "...after execution of the computer program has begun, automatically defining a user interface of the program by: reading a function description of a first function to be provided by the user interface on the fly at run time, the function description comprising instructions to handle user interface events; executing logic on the fly at run time to select an appearance description of a first appearance to be presented by the user interface; associating the function description and the appearance description on the fly at run time into an executable form; executing the executable form of the user interface to generate the user interface with the associated function description and appearance description; and executing logic to independently change one of the function description and the appearance description during program execution."

Thus, claim 1 is directed to automatically generating a user interface, without user interaction. As such, the function and appearance descriptions comprising the user interface can be automatically and independently changed by executing logic. Atkin et al., however, teach

changing the cultural profile of a user interface in response to a user initiated action – namely, the dragging and dropping of a locale change object onto a portion of an application. Further, claim 1 is directed to executing logic to select an appearance description, whereas Atkin et al. teach that the cultural profile is selected by a user. Therefore, Atkin et al. teach away from the subject matter recited in claim 1.

For example, the Office (Action of April 4, 2007 at page 3) asserts that Atkin et al. disclose (underlining added for emphasis)...

Executing logic on the fly at run time to select an appearance description of a first appearance to be presented by the user interface (see e.g. "...the locale change daemon takes the form of a drag and drop presentation manager application...When the daemon is invoked, a locale change may take place by simply dragging a locale change object and dropping it on top of an application..." in column 6, lines 20-26)

Atkin et al. fail to disclose the claimed subject matter.

For example, in its entirety Col. 6, lines 20-26 of Atkin et al. discloses (underlining added for emphasis)...

In the present invention, the locale change daemon takes the form of a drag and drop presentation manager application. The locale change daemon stays suspended until a user clicks on it with a mouse operation. When the daemon is invoked, a locale change may take place by simply dragging a locale change object and dropping it on top of an application.

Thus, Atkin et al. teach that the locale change daemon stays suspended until it is activated by a user clicking on it with a mouse operation. As such, the locale change daemon does not automatically define a user interface, as is claimed. For example, claim 1 recites that, after

execution of the computer program has begun, a user interface of the program is defined automatically by reading a function description of a first function...; executing logic on the fly at run time to select an appearance description...; and associating the function description and the appearance description on the fly at run time into an executable form. A locale change daemon that remains suspended until it is activated by a user does not automatically define a user interface. Rather, the locale change daemon performs no action until it is invoked by a user and then executes the user's command.

With respect to automatically defining a user interface, the Office (Action of April 4, 2007 at page 3) asserts...

automatically defining a user interface of the program by: reading a function description of a first function to be provided by the user interface on the fly at run time, the function description comprising instructions to handle user interface events (see e.g. "...locale object..." in column 5, lines 18-28)

The locale object disclosed by Atkin et al. does not teach automatically defining a user interface. Rather, Atkin et al. (Col. 5, lines 18-28) merely describe the local object...

The present invention includes a dynamic, object-oriented locale object that expands the currently defined notion of a locale by encapsulating localization information and localization procedures into a cohesive unit, thereby enabling an application to be internationalized. The locale object is then extended to allow for different cultural attributes, such as color, icons, dialogues, and menus. To further enhance the functionality of the extended locale object, the locale object also allows domain or application specific localization information to be contained therein.

Nothing in the cited portion of Atkin et al. discloses, teaches, or other wise suggests automatically defining a user interface.

Further, claim 1 recites executing logic on the fly at run time to select an appearance description. Atkin et al. do not disclose executing logic to select an appearance. To the contrary, Atkin et al. (Col. 6, lines 20-26) teach that a user selects a cultural profile by dragging a locale change object and dropping the locale change object on top of an application. Thus, the locale change daemon does not select an appearance. Rather, the locale change daemon simply implements the cultural profile selected by the user. Atkin et al. (Abstract) reinforce the user's role in performing the selection, stating (underlining added for emphasis)...

A user may dynamically change a program's cultural profile to a different cultural profile without having to reboot the system. The profile change may be accomplished through the use of a drag and drop interface.

Thus, Atkin et al. also do not disclose, teach, or suggest executing logic on the fly at run time to select an appearance description of a first appearance to be presented by the user interface, as is claimed.

For at least these reasons, claim 1 is allowable over Atkin et al. Claims 2-9 and 37-41 depend from claim 1. Therefore, dependent claims 2-9 and 37-41 are allowable for at least the reasons discussed with respect to claim 1.

Further, claims 21 and 45 include subject matter similar to that of claim 1. For example, claim 21 recites (underlining added for emphasis) "...read a function description of a first function to be provided by the user interface on the fly at run time,...; execute on the fly at run time logic to select an appearance description of a first appearance to be presented by the user

interface; associate the function description and the appearance description on the fly at run time; execute the user interface with the associated function description and appearance description;...” Similarly, claim 45 recites (underlining added for emphasis) “...reading a function description of a first function to be provided by the user interface on the fly at run time,...; executing on the fly at run time logic to select an appearance description of a first appearance to be presented by the user interface; associating the function description and the appearance description on the fly at run time into an executable form; executing the executable form of the user interface to generate the user interface with the associated function description and appearance description;...”

Therefore, claims 21 and 45 are allowable for at least the reasons discussed with respect to claim 1. Further, claims 22-29 and 42-44 depend from claim 21 and therefore are at least allowable based on claim 21. Additionally, claims 46-58 depend from claim 45 and therefore are at least allowable based on claim 45.

In addition, claim 2 recites (underlining added for emphasis) “...replacing the function description during program execution while providing a continuity of presentation.” Further, claim 3 recites (underlining added for emphasis) “...replacing the appearance description during program execution to present logic of the user interface with a different appearance.” Additionally, claim 1, from which claims 2 and 3 depend, recites (underlining added for emphasis) “...executing logic to independently change one of the function description and the appearance description during program execution.” Therefore, claims 1-3 recite changing one of the function description and the appearance description independently of the other.

With respect to claim 2, the Office (Action of April 4, 2007 at page 4) cites to Col. 7, lines 26-35 of *Atkin et al.* without further explanation. The cited portion of *Atkin et al.* discloses (underlining added for emphasis)...

To start a locale change, a user simply drags an icon flag from locale change daemon 54 and drops it on an application 44. When a drag-drop is initiated, locale change daemon 54 creates a binary file, and places the locale change information in this binary file, or locale change file, 58. When the drop takes place, application 44 retrieves the name of locale change file 58. Further details regarding this process are discussed below. Application 44 then reads file 58 and sends a message to extended locale object 48 telling it to change to a new locale Set locale.

Thus, *Atkin et al.* teach that, for every locale change, the locale change daemon is invoked and creates a locale change file. With respect to the locale change file 58, *Atkin et al.* (Col. 7, lines 40-44) further disclose (underlining added for emphasis)...

Next, as shown in FIG. 6, the drop is processed. When application 44 finishes reading locale change file 58, it signals to locale change daemon 54 that the drop has been processed. Locale change daemon then destroys locale change file 58.

Thus, *Atkin et al.* also teach that the locale change file to which the Office cites is used to process a locale change. Further, *Atkin et al.* teach that the locale change file is destroyed when the locale change is performed. Claim 1 recites that the function description comprises instructions to handle user interface events. *Atkin et al.* do not teach that the locale change file includes instructions to handle user interface events. Therefore, creating a locale change file is not equivalent to replacing the function description during program execution while providing a continuity of presentation, as is claimed.

Moreover, with respect to claim 3, the Office (Action of April 4, 2007 at page 4) asserts that Atkins et al. disclose...

replacing the appearance description during program execution to present logic of the user interface with a different appearance (column 6, lines 26-38).

The cited portion of Atkins et al. also relates to changing locale in response to a locale change object being dragged and dropped. For example, Atkins et al. (Col. 6, lines 26-33) disclose (underlining added for emphasis)...

The locale change is context sensitive. This means that if the locale change object is dropped onto the presentation space of the application, the data cultural profile for the application is changed to reflect the new cultural profile for the application's working data. If, on the other hand, the locale change object is dropped anywhere else on the application, the application's interaction cultural profile is changed to reflect the new cultural profile for the application.

Thus, the Office asserts that a locale change initiated by dropping a locale change object on an application replaces an appearance description.

A locale change process that necessarily changes the function description and the appearance description, as the Office asserts, cannot both (i) replace the function description during program execution while providing a continuity of presentation and (ii) replace the appearance description during program execution to present logic of the user interface with a different appearance. Such an assertion is logically inconsistent and is precluded by the plain language of the claims, which recite that the function description and appearance description are changed independently. Changing both the function description and the appearance description

simultaneously in response to a single locale change operation is not equivalent to changing the function description and the appearance description independently.

Therefore, in accordance with the Office's present assertion, claims 2 and 3 are allowable over *Atkin et al.* based on their own merits. Claims 22, 23, 46, and 47 include similar subject matter to claim 2 and 3, and were rejected under the same rationale. As such, claims 22, 23, 46, and 47 also are allowable over *Atkin et al.* for the same reasons as claims 2 and 3.

Claim 39 recites (underlining added for emphasis) "...selecting at least one of the function description and the appearance description based on a geographic location of a user of the computer program, wherein the function description is separate from the appearance description."

The Office (Action of April 4, 2007 at page 7) asserts that *Atkin et al.* disclose selecting at least one of the function description and the appearance description based on a geographic location of a user of the computer program at Col. 6, lines 20-26. *Atkin et al.*, however, fail to disclose the claimed subject matter.

Atkin et al. (Id.) disclose (underlining added for emphasis)...

In the present invention, the locale change daemon takes the form of a drag and drop presentation manager application. The locale change daemon stays suspended until a user clicks on it with a mouse operation. When the daemon is invoked, a locale change may take place by simply dragging a locale change object and dropping it on top of an application.

Thus, *Atkin et al.* do not disclose that a locale change takes place based on a geographic location of a user of the computer program, as is claimed. To the contrary, *Atkin et al.* teach that a locale

change is based on a user preference. For example, Atkin et al. (Col. 6, lines 34-39) disclose (underlining added for emphasis)...

A context sensitive locale change allows a user to interact with the application using one cultural profile, while working with data using a different cultural profile. For example, a user can work with a spreadsheet using an Arabic toolbar, commands, etc., while the spreadsheet actually contains Japanese currency.

Therefore, Atkin et al. teach that the locale change is performed independent of the geographic location of the user and allows a user to specify any desired cultural profile or combination of cultural profiles.

For at least these reasons, claim 39 also is allowable over Atkin et al. based on its own merits. Further, claims 42 and 54 contain subject matter similar to that of claim 39 and are therefore also allowable over Atkin et al. for the same reasons.

Rejection Under 35 U.S.C. §103(a)

Claim 58 stands rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Atkin et al. in view of Sanna, et al., Using Windows NT Workstation 4.0, 1997 ("Sanna").

Claim 45, from which claim 58 depends, recites (underlining added for emphasis) "a presentation device to present a user interface...." Further, claim 58 recites (underlining added for emphasis) "...wherein the presentation device comprises a telephone." Therefore, claim 58 recites a telephone that presents a user interface.

The Office (Action of April 4, 2007 at page 9) concedes that Atkin et al. do not teach a presentation device that comprises a telephone. The Office (Id.), however, asserts that

(underlining added for emphasis) “Sanna teaches that a telephone connects to a PC system and utilizes the PC monitor to present a user interface associated with the telephone (‘Using Phone Dialer’, pg. 442, par. 6 to par. 7).” Further, the Office (*Id.*) asserts (underlining added for emphasis)...

It would have been obvious to one having ordinary skill in the computer art at the time of the invention was made to modify the system disclosed by Atkin to include that the presentation device comprises a telephone using the teaching of Sanna. The modification would be obvious because one of ordinary skill in the art would be motivated to give a user an extra option to present the user interface.

The proposed combination of Atkin et al. and Sanna fails to disclose, teach, or suggest the claimed subject matter.

Sanna does not disclose, teach, or suggest using a telephone as a presentation device to present a user interface. To the contrary, Sanna teaches using a display to present a user interface associated with a telephone.

Sanna (p. 442, ¶6) indicates that Windows NT 4.0 includes a Phone Dialer application that can be used to dial a telephone connected to the host computer. Further, Sanna (pp. 442-444) states that the Phone Dialer application includes an interface displayed on a monitor. Thus, Sanna teaches displaying a user interface on a monitor – not presenting a user interface through a telephone.

Further, Sanna (*Id.*) discloses that the interface displayed on the monitor can be used to dial the telephone or to store telephone numbers in a speed-dial. Sanna does not, however, disclose, teach, or suggest that the telephone coupled to the processor presents the user interface, as is claimed.

To the contrary, Sanna teaches that the Phone Dialer application, which is displayed to the user via a visual display (e.g., monitor), allows a user to dial an associated telephone. Moreover, there is no motivation to combine the cited references because Sanna discloses only the use of a display for presenting a user interface, which also is disclosed in Atkin et al.

For at least these reasons, claim 58 is allowable over the proposed combination of Atkin et al. and Sanna based on its own merits.

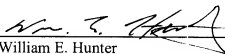
Concluding Comments

The foregoing comments made with respect to the positions taken by the Examiner are not to be construed as acquiescence with other positions of the Examiner that have not been explicitly contested. Accordingly, the above arguments for patentability of a claim should not be construed as implying that there are not other valid reasons for patentability of that claim or other claims.

In view of the above remarks, claims 1-9, 21-29, and 37-58 should be in condition for allowance, and a formal notice of allowance is respectfully requested. Please apply any charges or credits to deposit account 06-1050.

Respectfully submitted,

Date: July 5, 2007



William E. Hunter
Reg. No. 47,671

Fish & Richardson P.C.
PTO Customer No. 21876
Telephone: (858) 678-5070
Facsimile: (858) 678-5099
10752090.doc